

BY EXPRESS MAIL NO. EL387309184US  
Attorney Docket No. KOIK-P9784

MENU

SEARCH

INDEX

1/1



JAPANESE PATENT OFFICE

## PATENT ABSTRACTS OF JAPAN

(11)Publication number: 06282440

(43)Date of publication of application: 07.10.1994

(51)Int.Cl.

G06F 9/45

(21)Application number: 05066947

(71)Applicant:

SONY CORP

(22)Date of filing: 25.03.1993

(72)Inventor:

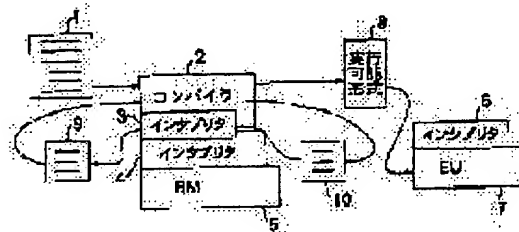
KITATANI YOSHIMICHI

(54) PROGRAM EDITING METHOD, PROGRAM EDITING AND EXECUTING METHOD, AND COMPUTER SYSTEM

(57)Abstract:

**PURPOSE:** To provide a processing system which edits a program and the program editing method which can efficiently edit machine codes that can be executed on a processing system different from the processing system.

**CONSTITUTION:** A program processing system RM which has a compiler 2 for editing the source program 1 and generates the machine codes and a program processing system EU which executes the generated machine codes are provided. The processing system RM is provided with an interpreter 4 and an interpreter 3 and a compiler 2 which operate on the interpreter 4. The interpreter 3 simulates the operation of an interpreter 6 which operates on the processing system EU. The compiler 2 reads in the source program 1 of a LISP language process and generates a program 10 to be run on the interrupter 3. The interpreter 3 executes the program 10 instead of the interpreter 6 and offers program information which can not be execute to the compiler 2 as a program 9. The compiler 2 further edits the program 9. The interpreter 4 executes the generated machine codes 8 on the processing system EU.



LEGAL STATUS

[Date of request for examination]  
[Date of sending the examiner's decision of rejection]  
[Kind of final disposal of application other than the  
examiner's decision of rejection or application converted  
registration]  
[Date of final disposal for application]  
[Patent number]  
[Date of registration]  
[Number of appeal against examiner's decision of rejection]  
[Date of requesting appeal against examiner's decision of  
rejection]  
[Date of extinction of right]

Copyright (C); 1998 Japanese Patent Office

---

[MENU](#)

[SEARCH](#)

[INDEX](#)

(19)日本国特許庁(JP)

(12)公開特許公報(A)

(11)特許出願公開番号

特開平6-282440

(43)公開日 平成6年(1994)10月7日

(51)Int.Cl.<sup>5</sup>

G 0 6 F 9/45

識別記号

庁内整理番号

9292-5B

F I

G 0 6 F 9/44

技術表示箇所

3 2 0 A

審査請求 未請求 請求項の数 8 O L (全 9 頁)

(21)出願番号

特願平5-66947

(22)出願日

平成5年(1993)3月25日

(71)出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72)発明者 北谷 義道

東京都品川区北品川6丁目7番35号 ソニ

ー株式会社内

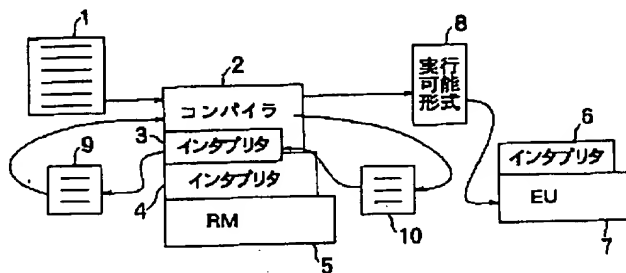
(74)代理人 弁理士 佐藤 隆久

(54)【発明の名称】 プログラム編集方法、プログラム編集・実行方法およびコンピュータシステム

(57)【要約】

【目的】 プログラム編集する処理系と、この処理系とは異なる処理系で実行可能な機械語コードを効率よく編集可能なプログラム編集方法を提供する。

【構成】 ソースプログラム11を編集して機械語コードを生成するコンパイラ1を有するプログラム処理系RMと、生成された機械語コードを実行するプログラム処理系EUと有する。処理系RMには、インタプリタ4、このインタプリタ4の上で動作するインタプリタ3およびコンパイラ2が設けられている。インタプリタ3は処理系EUで動作するインタプリタ6の動作を模擬する。コンパイラ2はLISP言語処理のソースプログラム1を読み込んで、インタプリタ3で動作させるプログラム10を生成する。インタプリタ3はインタプリタ6に代わってプログラム10を実行して、実行できないプログラム情報をプログラム9としてコンパイラ2に提供する。コンパイラ2はそのプログラム9をさらに編集する。インタプリタ4は生成された機械語コード8を処理系EUにおいて実行する。



## 【特許請求の範囲】

【請求項1】プログラム実行の前に、実行されるプログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムについて、

そのソースプログラムが実行されるプログラム処理系と別のプログラム処理系で動作し、  
プログラムが実行されるプログラム処理系の機能を擬似的に実現する処理機能を有し、  
前記擬似実行によって前処理を行なうことを特徴とするプログラム編集方法。

【請求項2】機械語コードに変換される際に、実行されるプログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムについて、  
擬似実行による前処理を行ないながら機械語コードに変換することを特徴とする請求項1記載のプログラム編集方法。

【請求項3】ソースプログラムの実行とは異なったプログラム処理系で前処理が実行され、  
プログラムが実行されるプログラム処理系に対応した次段階のプログラム言語で記述されたプログラムを作成することを特徴とする請求項1記載のプログラム編集方法。

【請求項4】前記ソースプログラムがLISP言語で生成されている、請求項1～3いずれか記載のプログラム編集方法。

【請求項5】第1のプログラム処理系で動作するプログラム編集処理プログラムと、第1のインタプリタと第2のインタプリタを有し、  
第1のプログラム処理系に接続された第2のプログラム処理系で動作する第3のインタプリタを有し、  
前記プログラム編集処理プログラムは、プログラム実行の前に、実行されるプログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムに編集し、第1のインタプリタ言語および第2のプログラム処理系で実行される機械語コードに変換する機能を有し、  
第2のインタプリタは、前記プログラム編集プログラムが編集されて生成された第1のインタプリタ言語を、第2のプログラム処理系における第3のインタプリタの動作を模擬して実行して第2のインタプリタ言語を生成し、  
第3のインタプリタは該第1のインタプリタ言語を処理してプログラム編集情報を前記プログラム編集処理プログラムに提供し、  
前記プログラム編集処理プログラムが該プログラム編集情報を編集して、第2のプログラム処理系における第3のインタプリタで実行される前記機械語コードを生成し、

第3のインタプリタは該機械語コードを前記第2のプログラム処理系において実行するプログラム編集・実行方法。

【請求項6】前記ソースプログラムがLISP言語で生成されている、請求項5記載のプログラム編集・実行方法。

【請求項7】第1のプログラム処理系と該第1のプログラム処理系に接続された第2のプログラム処理系とを有し、

10 第1のプログラム処理系は、プログラム編集処理プログラムと、第1のインタプリタと第2のインタプリタを有し、

第2のプログラム処理系は第3のインタプリタを有し、  
前記プログラム編集処理プログラムは、プログラム実行の前に、実行されるプログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムに編集し、第1のインタプリタ言語および第2のプログラム処理系で実行される機械語コードに変換する機能を有し、

20 第2のインタプリタは、前記プログラム編集プログラムが編集をされて生成された第1のインタプリタ言語を、第2のプログラム処理系における第3のインタプリタを模擬して実行して第2のインタプリタ言語を生成し、  
第3のインタプリタは第2のインタプリタ言語を処理してプログラム編集情報を前記プログラム編集処理プログラムに提供し、

前記プログラム編集処理プログラムが該プログラム編集情報を編集して、前記機械語コードを生成し、  
第3のインタプリタは該機械語コードを前記第2のプログラム処理系において実行するコンピュータシステム。

30 【請求項8】前記ソースプログラムがLISP言語で生成されている、請求項7記載のコンピュータシステム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、デジタル計算機におけるプログラムを編集する方法に関するものであり、特に、実行させる処理系とは異なる処理系で編集を行なうプログラム編集方法、プログラム編集・実行方法、および、これらを行うコンピュータシステムに関する。

40 【0002】

【従来の技術】従来から、C言語などの高級プログラミング言語で記述されたソースプログラムを機械語に変換するといったコンパイラと、生成された機械語を読み取り実行する処理系とは別の処理系で動作させるようなコンパイラであるクロスコンパイラとを用いるプログラム編集方法が知られており、実際に使用されている。以下、クロスコンパイラを使用し、高級プログラム言語で記述されたソースプログラムを下位言語に変換する、従来のプログラム編集方法について述べる。

50 【0003】図5は従来のクロスコンパイラのデータの

流れを示した図である。図5においてソースプログラム11はC言語で記述されたプログラムである。コンパイラ12はプログラム処理系(A)13上で動作し、ソースプログラム11を読み込んでコンピュータで実行可能な機械語コード14を生成する。この機械語コード14はプログラム処理系(B)15に読み込まれて実行される。

【0004】また、クロスコンパイルと呼ばれるプログラム編集方法とは別に、従来から、図6に示したように、1つの処理系の中にソースプログラム11を読み込み、機械語コード14に変換しながら随時プログラムの実行を行なうインタプリタ16と、ソースプログラム11から機械語コード14に一括してプログラムを編集するコンパイラ12が1つの処理系内に同時に存在し、コンパイル時にインタプリタ16にコンパイル処理の一部として実行されるプログラムの変換処理を記述できるプログラミング言語と、ソースプログラム中の指定された処理をコンパイル中にインタプリタ16に実行させるようなプログラム編集方法が知られており、実際に使用されている。

【0005】図6においてソースプログラム11はLISPで記述されたプログラムである。コンパイラ12はLISPインタプリタ16上で動作し、ソースプログラム11を読み込んで実行可能な機械語コード14を生成する。コンパイラ12はソースプログラム11中にコンパイル時の実行を指定するプログラムを見つけると、プログラム17としてインタプリタ16に渡し実行させる。インタプリタ16はその結果生じる出力を新たなプログラム18としてコンパイラ12に再び戻す。コンパイラ12はプログラム18を受け取るとそれを機械語コード14に変換する。

【0006】

【発明が解決しようとする課題】しかしながら、図6を参照して上述した従来のプログラム編集方法では、一つの処理系の中にインタプリタ16とコンパイラ12が同時に存在し、コンパイル時にインタプリタ16にコンパイル処理の一部として実行させる処理を記述できるプログラミング言語で書かれたソースプログラム11を、そのソースプログラム11から生成された機械語コード14が実行されるプログラム処理系(図示せず)とは別の処理系、この例ではフレーム処理系(A)13で、機械語コード5に変換することはできない。そのため、従来のプログラム編集方法では、プログラム処理系を開発する場合、プログラム処理系がコンパイラを動作させるだけ充分な機能を持つ前にプログラム処理系の機能をコンパイラによって作成し、効率的に開発を行なうことが出来ないという問題が生じる。

【0007】また、従来のプログラム編集方法では、より高速な処理系上のコンパイラを用いて機械語コードを作成し、より効率的にプログラム開発を行なうことがで

きないという問題が生じる。

【0008】したがって、本発明は、プログラムを編集するプログラム処理系とは異なるプログラム処理系で動作可能な機械語コードを生成可能なプログラム編集方法を提供することを目的とする。また本発明は、効率良くプログラム開発可能なフレーム編集方法を提供することを目的とする。

【0009】さらに本発明は、上述したプログラム編集およびその実行が可能なコンピュータシステムを提供することを目的とする。

【0010】

【課題を解決するための手段、および、作用】上述した問題を解決するため、本発明のプログラム編集方法においては、プログラム実行の前に、実行されるプログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムについて、そのソースプログラムが実行されるプログラム処理系と別のプログラム処理系で動作し、プログラムが実行されるプログラム処理系の機能を擬似的に実現する機能を有し、擬似実行によって前処理を行なうようにした。

【0011】また、本発明のプログラム編集方法においては、機械語コードに変換される際に、プログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムについて、疑似実行による前処理を行ないながら機械語コードに変換するようにした。

【0012】さらに、本発明のプログラム編集方法においては、ソースプログラムの実行とは異なったプログラム処理系(計算機)で動作し、プログラムが実行される処理系の機能を疑似的に実現する機能を有し、疑似実行によって前処理を行ない、プログラムが実行されるプログラム処理系(計算機)に対応した次段階のプログラム言語で記述されたプログラムを作成するようにした。

【0013】上記ソースプログラムとしては、好適には、LISP言語で記載されたプログラムである。

【0014】また本発明によれば、第1のプログラム処理系で動作するプログラム編集処理プログラムと、第1のインタプリタと第2のインタプリタを有し、第1のプログラム処理系に接続された第2のプログラム処理系で動作する第3のインタプリタを有し、前記プログラム編集処理プログラムは、プログラム実行の前に、実行されるプログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムに編集し、第1のインタプリタ言語および第2のプログラム処理系で実行される機械語コードに変換する機能を有し、第2のインタプリタは、前記プログラム編集プログラムが編集されて生成された第1のインタプリタ言語を、第2のプログラム処理系における第3のインタプリタの動作を模擬して実行して第2

のインタプリタ言語を生成し、第3のインタプリタは該第1のインタプリタ言語を処理してプログラム編集情報を前記プログラム編集処理プログラムに提供し、前記プログラム編集処理プログラムが該プログラム編集情報を編集して、第2のプログラム処理系における第3のインタプリタで実行される前記機械語コードを生成し、第3のインタプリタは該機械語コードを前記第2のプログラム処理系において実行するプログラム編集・実行方法が提供される。

【0015】さらに、本発明によれば、第1のプログラム処理系と該第1のプログラム処理系に接続された第2のプログラム処理系とを有し、第1のプログラム処理系は、プログラム編集処理プログラムと、第1のインタプリタと第2のインタプリタを有し、第2のプログラム処理系は第3のインタプリタを有し、前記プログラム編集処理プログラムは、プログラム実行の前に、実行されるプログラム自身を編集し変換する前処理を、自身で記述できる言語要素を備えたプログラミング言語で記述されたソースプログラムに編集し、第1のインタプリタ言語および第2のプログラム処理系で実行される機械語コードに変換する機能を有し、第2のインタプリタは、前記プログラム編集プログラムが編集をされて生成された第1のインタプリタ言語を、第2のプログラム処理系における第3のインタプリタを模擬して実行して第2のインタプリタ言語を生成し、第3のインタプリタは第2のインタプリタ言語を処理してプログラム編集情報を前記プログラム編集処理プログラムに提供し、前記プログラム編集処理プログラムが該プログラム編集情報を編集して、前記機械語コードを生成し、第3のインタプリタは該機械語コードを前記第2のプログラム処理系において実行するコンピュータシステムが提供される。

#### 【0016】

【実施例】図面を参照して本発明の実施例について説明する。図1は、本発明のプログラム編集方法が適用されるプロセッサエレメントPE（コンピュータシステム）の概略構成図である。図1（A）は単独のプロセッサエレメントPEの構成を示す図であり、図1（B）は複数のプロセッサエレメントPEが相互に接続されて大規模コンピュータシステムを構成する図を示す。

【0017】図1（A）に示したプロセッサエレメントPEは、LISPなどの記号言語を処理する記号処理用計算機エバリュエータEUと入出力や記憶管理に使われる計算機RMと主記憶MMおよび二次記憶装置とを有するストレージ（記憶装置）から構成されている。記号処理用計算機エバリュエータEUに対して、入出力や記憶管理に使われる計算機RMおよびストレージが一種のリソース（資源）として扱われている。

【0018】図1（B）は、各プロセッサエレメントPE内の入出力や記憶管理に使われる計算機RMが、LAN（Local Area Network）などの相互接続系（インター

リソースコネクション）を介して、相互に接続されて、複数のプロセッサエレメントPEが全体として、大規模コンピュータシステムを構成している。

【0019】図2は図1（A）に示されたプロセッサエレメント（PE）の構成の詳細を示す図である。本発明のプログラム編集方法は、図2に示すような記号処理用計算機エバリュエータEUと入出力や記憶管理に使われる計算機（RM）からなる並列計算機ネットワークのプロセッサエレメントPEに適用される。図5および図6に対応させると、入出力や記憶管理に使われる計算機RMがプログラム処理系Aに対応し、入出力や記憶管理に使われる計算機RMがプログラム処理系Bに対応している。

【0020】記号処理用計算機エバリュエータEUは、データ幅32ビットのVLIW（Very Long Instruction Word）制御のプロセッサであり、ユーザ定義の関数やメソッドのコンパイルされたコードを高速実行するためのインストラクションキャッシュ（64ビット×64Kワード）を持つ。また、記号処理用計算機エバリュエータEUは、データのフィールド演算を高速に実現するためのマスカ（Masker）／算術論理処理ユニット

（ALU）を持ち、関数やメソッドの呼び出しを高速化するためのデータフィールド値による多方向分岐機能を持ち、さらに多方向分岐のためのルックアップテーブルと記号処理用計算機エバリュエータEUのシステムスタックのための64Kワードの制御用メモリを持つ。さらに記号処理用計算機エバリュエータEU、記号処理計算を実行する際のフレームスタックとレジスタファイルのための64Kワードのローカルメモリを持つ。記号処理用計算機エバリュエータEUは、入出力や記憶管理に要する演算は、入出力や記憶管理に使われる計算機RMにその実行を要求する。

【0021】入出力や記憶管理に使われる計算機RMは、記憶管理、デバイス管理、プロセス管理、ファイル管理、入出力、他のプロセッサエレメントPEとの通信の機能を記号処理用計算機エバリュエータEUに提供する。入出力や記憶管理に使われる計算機RMには、ワークステーションWSを使用して、ワークステーションWSのXウィンドウ・システム上にアプリケーション開発時および実行時のインタラクティブなヒューマンインタフェース環境を実現する。そのため、入出力や記憶管理に使われる計算機RMは関数定義やメソッド定義のインクリメンタルな修正・コンパイル、エラー状態からの復帰・実行再開機能を持つ。さらに入出力や記憶管理に使われる計算機RMは、アプリケーションの実行時にユーザとの動的な対話ができるような入出力機能を提供する。入出力や記憶管理に使われる計算機RMは、主記憶MM上の記号処理データの高速アクセスのためのポイント・マニピレータ（Pointer Manipulator）を持つ。

50 入出力や記憶管理に使われる計算機RMは、並列処理時

に他のプロセッサエレメントPEとの通信を行なうための通信インタフェースを装着可能である。

【0022】記憶装置（ストレージ）のうち主記憶MMは、記号処理用計算機エバリュエータEUからも直接アクセス可能なデュアルポートメモリである。主記憶MMでは、排他的アクセス、記号処理演算で有効な参照カウンタなどの機能アクセスを提供する。主記憶MMは、これらの制御情報と2ワードのポインタを表現するため、80ビット×256Kワードの構成となっている。

【0023】記憶装置のうち二次記憶は、ワークステーションWSのディスクシステムを使用する。

【0024】記号処理用計算機エバリュエータEUと入出力や記憶管理に使われる計算機RMとは、入出力や記憶管理に使われる計算機インターフェースRM-I/Fで接続され、記号処理用計算機エバリュエータEUから入出力や記憶管理に使われる計算機RMに、また入出力や記憶管理に使われる計算機RMから記号処理用計算機エバリュエータEUに機能データを送出する。また、記号処理用計算機エバリュエータEUからメインメモリインターフェースMM-I/Fを介して情報データが送出され、主記憶MMと入出力や記憶管理に使われる計算機RMに送出される。

【0025】入出力や記憶管理に使われる計算機インターフェースRM-I/F、および、主記憶インターフェースMM-I/Fは、記号処理用計算機エバリュエータEU内の内部バスIBUSを介して算術論理処理ユニットALUに接続される。また、入出力や記憶管理に使われる計算機インターフェースRM-I/Fは、入出力や記憶管理に使われる計算機RM内のエバリュエータインターフェースEU-I/Fを介してワークステーションWSに接続される。

【0026】図3は本発明の実施例のコンパイラのソフトウェア構成を表す図である。入出力や記憶管理に使われる計算機(RM)5が、図5および図6におけるプログラム処理系Aに対応し、記号処理用計算機エバリュエータ(EU)7がプログラム処理系Bに対応している。入出力や記憶管理に使われる計算機(RM)5には、コンパイラ2、第1のインタプリタ3および第2のインタプリタ4が入出力や記憶管理に使われる計算機(RM)5の上で動作し、実行可能な機械コード8を生成する。記号処理用計算機エバリュエータ(EU)7が実行可能な機械語コード8を実行させる第3のインタプリタ6を内蔵している。

【0027】ソースプログラム1は、LISPで記述されたソースプログラムである。インタプリタ4は入出力や記憶管理に使われる計算機RM5上で動作しするインタプリタで、コンパイラ2およびインタプリタ3はこのインタプリタ4上で動作する。入出力や記憶管理に使われる計算機(RM)5は、図2に示した計算機(RM)である。インタプリタ3は、記号処理用計算機エバ

リュエータEU上のインタプリタ6の動作を疑似的に実行するインタプリタであり、コンパイラ2が処理中に呼び出でされる。コンパイラ2は、インタプリタ3、インタプリタ4とともにLISPで記述されたソースプログラム1の解析を行ない、記号処理用計算機エバリュエータ(EU)7において実行可能な形式の機械語コード8を生成する。機械語コード8は記号処理用計算機エバリュエータ(EU)7に読み込まれ実行される。記号処理用計算機エバリュエータEUは、図2のエバリュエータEUである。

【0028】図4は本発明のコンパイラのプログラム編集方法およびその実行を示すフローチャートである。入出力や記憶管理に使われる計算機(RM)5上のコンパイラ2は、ステップ01(S01)において起動されると、ステップ02(S02)においてソースプログラム1からプログラムの一部を取り込み、プログラムがある限り以下の処理、ステップ04~05、06~07を繰り返す。プログラム処理が終了すると、ステップ03(S03)において、プログラム編集処理が終了する。

【0029】S04において取り込んだ入力が入タプリタ3に前処理の実行を指定した記述である場合、コンパイラ2は、そのプログラムの部分10を第1のインタプリタ言語として、記号処理用計算機エバリュエータEUにおけるインタプリタ6の模擬動作を行うインタプリタ3に渡す。S04において取り込んだ入力が入タプリタ3に前処理の実行を指定した記述でない場合、コンパイラ2は、入力されたプログラム1に対して機械語コード8を生成する。インタプリタ3は、S06においてそのプログラムを実行し、その結果として編集用プログラム9を生成する。コンパイラ2はS07において編集用プログラム9を新たに入力として取り込み、その編集用プログラム9に対してさらにコンパイル処理を行なう。

【0030】このように、本発明の実施例によれば、プログラムが実行される処理系である記号処理用計算機エバリュエータEUとは別の処理系である入出力や記憶管理に使われる計算機RMでプログラミングを編集することを可能とし、プログラム処理系開発の初期段階における開発効率を向上させることが出来る。また、本発明のプログラム編集方法によれば、より高速な計算機上で目的プログラムの編集を行ない、開発の効率を向上させることが出来る。

【0031】さらに本発明の実施例によれば、疑似実行による変換を行ないながら機械語に変換することにより、プログラムが実行される処理系とは別の処理系でプログラムを編集することを可能とし、処理系開発の初期段階における開発効率を向上させることが出来る。あるいは、より高速な計算機上で目的プログラムの編集を行ない、プログラム開発の効率を向上させることが出来る。

【0032】上述した実施例は、図1（A）に示した単独のプロセッサエレメントPEにおけるプロセッサエレメントPE編集処理およびその実行を例示したが、本発明のプログラム編集方法は、図1（B）に示したコンピュータシステムにおいても、適用できる。たとえば、第1のプロセッサエレメントPE内の入出力や記憶管理に使われる計算機RMをプログラム処理系Aとし、この入出力や記憶管理に使われる計算機RMにおいて上述したプログラム編集処理を行い、その結果としての機械コードをLANなどで接続された第2のプロセッサエレメントPE内の記号処理用計算機エバリュエータEUまたは入出力や記憶管理に使われる計算機RMにおいて実行させることもできる。

#### 【0033】

【発明の効果】本発明のプログラム編集方法によれば、プログラム実行の前処理を自身で記述できる言語要素を備えた高級プログラミング言語で記述されたソースプログラムについて、そのソースプログラムが実行される処理系と別の処理系で動作し、プログラムが実行される処理系の機能を疑似的に実現する機能を有し、疑似実行によってソースプログラムの前処理を行ってから機械語に変換することにより、プログラムが実行される処理系とは別の処理系でプログラミングを編集することを可能とし、処理系開発の初期段階における開発効率を向上させることが出来る。

【0034】また、本発明のプログラム編集方法によれば、より高速な計算機上で目的プログラムの編集を行ない、開発の効率を向上させることが出来る。

【0035】さらに本発明のプログラム編集方法によれば、機械語コードに変換される際に自身で記述された機能を実行する必要がある高級プログラミング言語で記述されたソースプログラムについて、そのソースプログラムが実行される処理系と別の処理系で動作し、プログラムが実行される処理系の機能を疑似的に実現する機能を有し、疑似実行による変換を行ないながら機械語コードに変換することにより、プログラムが実行される処理系とは別の処理系でプログラムを編集することを可能とし、処理系開発の初期段階における開発効率を向上させることが出来る。あるいは、より高速な計算機上で目的プログラムの編集を行ない、プログラム開発の効率を向

上させることが出来る。

【0036】本発明によれば、プログラム編集と実行を連続的に行うプログラム編集・実行方法が提供できる。

【0037】さらに本発明によれば、上記プログラム編集・実行を行うコンピュータシステムを提供できる。

#### 【図面の簡単な説明】

【図1】本発明が適用される計算機ネットワークの接続形態を示す図であり、（A）は、プロセッサエレメントPE単体の構成図であり、（B）は複数のプロセッサエレメントPEが大規模コンピュータシステムを構成する図である。

【図2】図1に示したプロセッサエレメントPEの詳細構成を示す図である。

【図3】本発明のコンパイラのソフトウェア構成を表す図である。

【図4】本発明のプログラム編集・実行をの処理のフローチャートである。

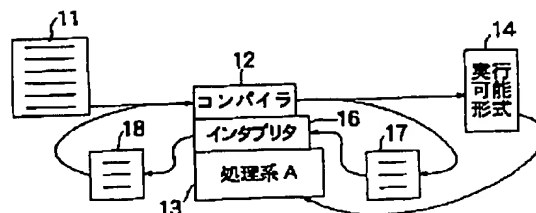
【図5】従来のクロスコンパイラのソフトウェア構成を示す図である。

【図6】従来のインタプリタとコンパイラのソフトウェア構成を示す図である。

#### 【符号の説明】

- 1・・・プログラム編集対象のプログラム
- 2・・・コンパイラ
- 3・・・記号処理用計算機エバリュエータのインタプリタの動作を模擬するインタプリタ
- 4・・・入出力や記憶管理に使われる計算機RMの上で動作する基本的なインタプリタ
- 5・・・入出力や記憶管理に使われる計算機RM
- 6・・・記号処理用計算機エバリュエータEUの上で動作するインタプリタ
- 7・・・記号処理用計算機エバリュエータEU
- 8・・・記号処理用計算機エバリュエータEUで実行可能な機械語コード
- 9・・・インタプリタ4で生成された編集用プログラムRM
- 10・・・入出力や記憶管理に使われる計算機（プログラム処理系A）
- 11・・・記号処理用計算機エバリュエータ（プログラム処理系B）

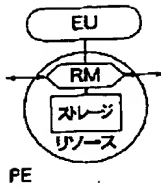
【図6】



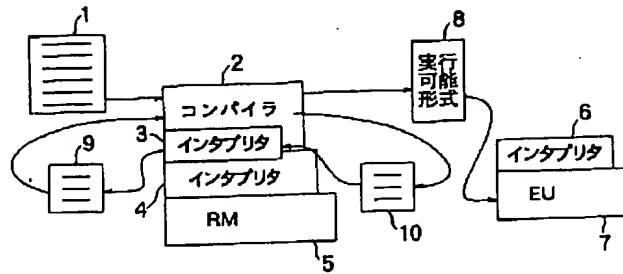


【図1】

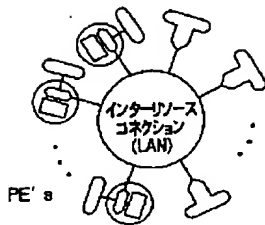
(A)



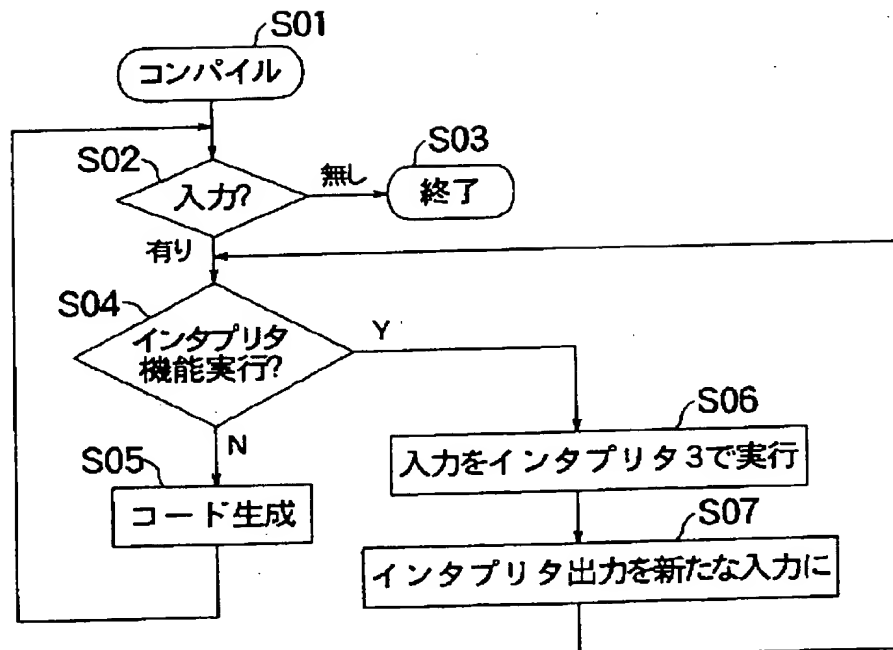
【図3】



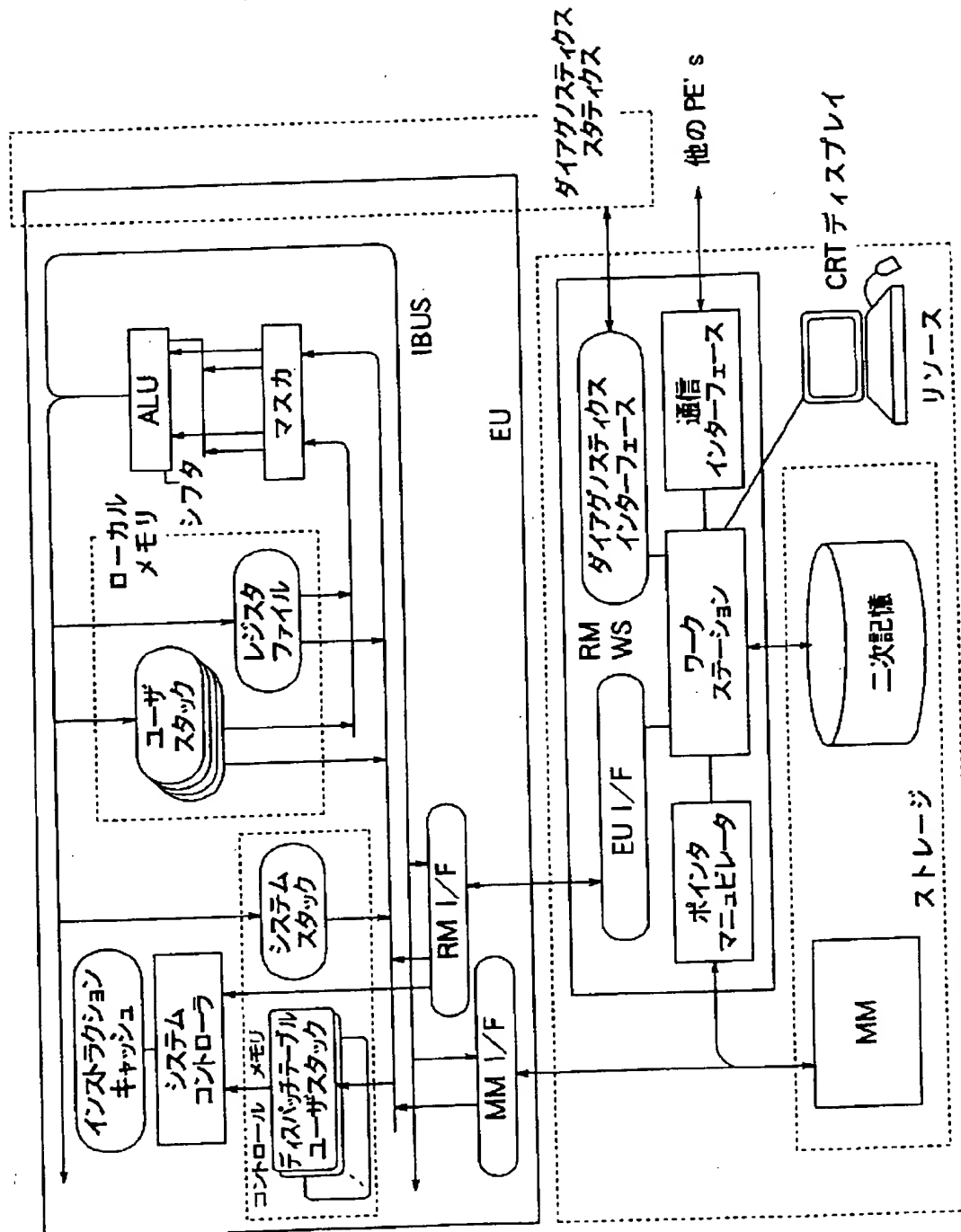
(B)



【図4】



【図2】



```

graph LR
    11[11] --> 12[12 コンパイラ]
    12 --> 14[14 実行可能形式]
    14 --> 15[15 処理系 B]
    13[13 処理系 A]
  
```